### I.2. Nguyên tắc hoạt động Website

### 

Hình ảnh đầu tiên mô tả tổng quan cách một website hoạt động, từ người dùng đến máy chủ và những thành phần liên quan.

1. **Users (clients) / Web browser (Người dùng / Trình duyệt web):**

· Người dùng truy cập website thông qua các trình duyệt web (Chrome, Firefox, Edge, Safari...).

· Trình duyệt web là ứng dụng gửi yêu cầu đến máy chủ web.

2. **Internet:**

* Là mạng lưới toàn cầu kết nối các máy tính và máy chủ.
* Khi bạn nhập một URL (Uniform Resource Locator - địa chỉ web) vào trình duyệt, yêu cầu của bạn sẽ đi qua Internet.

3.**Domain name (Tên miền) và URL (Địa chỉ web):**

* · **URL:** Là địa chỉ cụ thể của một trang web (ví dụ: https://www.example.com/trang-chu).
* **Domain name:** Là phần tên dễ nhớ của địa chỉ web (ví dụ: example.com).

4.**DNS server (Máy chủ DNS - Domain Name System):**

* Khi người dùng nhập tên miền vào trình duyệt, trình duyệt sẽ gửi yêu cầu đến máy chủ DNS.
* Máy chủ DNS có nhiệm vụ "dịch" tên miền thành địa chỉ IP (Internet Protocol - địa chỉ số của máy chủ). Ví dụ, nó sẽ dịch example.com thành 192.0.2.1.
* Đây là bước cực kỳ quan trọng vì Internet hoạt động dựa trên địa chỉ IP, không phải tên miền.

5.**WWW Servers (Máy chủ WWW / Web Servers):**

* Sau khi có được địa chỉ IP từ DNS, trình duyệt sẽ gửi yêu cầu HTTP/HTTPS đến máy chủ web có địa chỉ IP đó.
* Máy chủ web là máy tính chuyên dụng lưu trữ nội dung website (file HTML, CSS, JavaScript, hình ảnh, video...).
* Khi nhận được yêu cầu, máy chủ web sẽ tìm kiếm nội dung tương ứng và gửi trả về trình duyệt của người dùng.

6. **Website content (Nội dung website):**

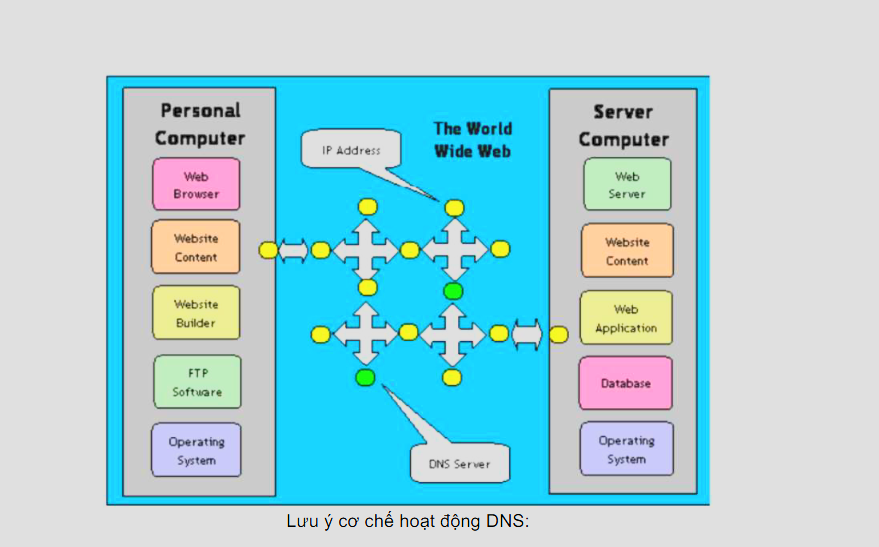
· Là tất cả các tài nguyên tạo nên một trang web, được lưu trữ trên máy chủ web.

7. **Authors / Web builder (Tác giả / Người xây dựng web):**

· Đây là những người tạo ra nội dung và cấu trúc của website.

* Họ sử dụng các công cụ, phần mềm để xây dựng và quản lý website.

**Tóm tắt luồng hoạt động:** Người dùng nhập URL vào trình duyệt -> Trình duyệt hỏi DNS server để lấy IP của tên miền -> DNS server trả về IP -> Trình duyệt gửi yêu cầu đến Web server theo IP đó -> Web server gửi nội dung website về trình duyệt -> Trình duyệt hiển thị nội dung cho người dùng.



**1. Personal Computer (Máy tính cá nhân của người dùng):**

* **Web Browser:** Trình duyệt web, ứng dụng để xem website.
* **Website Content:** (Trong ngữ cảnh này, có thể hiểu là các file tạm thời hoặc bộ nhớ cache mà trình duyệt lưu lại từ website).
* **Website Builder:** Các phần mềm dùng để thiết kế web (ví dụ: Adobe Dreamweaver, Visual Studio Code...).
* **FTP Software:** Phần mềm truyền tải file (File Transfer Protocol) dùng để tải file từ máy tính cá nhân lên máy chủ hoặc ngược lại.
* **Operating System:** Hệ điều hành của máy tính (Windows, macOS, Linux...).

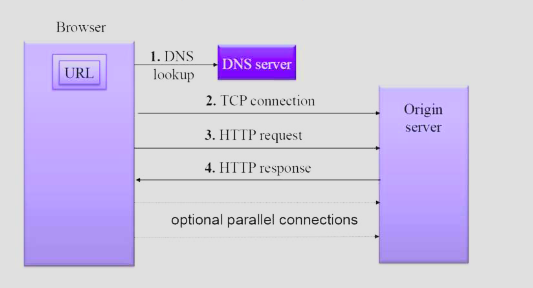
**2. The World Wide Web (Mạng lưới toàn cầu - Internet):**

* **IP Address:** Địa chỉ IP, được dùng để định danh các thiết bị trên mạng.
* **DNS Server:** Như đã giải thích ở trên, có nhiệm vụ chuyển đổi tên miền thành địa chỉ IP.
* Các mũi tên và đường kẻ thể hiện luồng dữ liệu và kết nối giữa các thành phần.

**3. Server Computer (Máy chủ):**

* **Web Server:** Phần mềm chạy trên máy chủ, có nhiệm vụ nhận yêu cầu từ trình duyệt và gửi trả nội dung website. (Ví dụ: Apache, Nginx, IIS).
* **Website Content:** Toàn bộ file của website (HTML, CSS, JS, hình ảnh...) được lưu trữ trên máy chủ.
* **Web Application:** Các ứng dụng chạy trên máy chủ để xử lý các logic phức tạp, tương tác với cơ sở dữ liệu (ví dụ: ứng dụng viết bằng PHP, Python, Java, Node.js...).
* **Database:** Nơi lưu trữ dữ liệu động của website (ví dụ: thông tin người dùng, bài viết, sản phẩm...).
* **Operating System:** Hệ điều hành của máy chủ (ví dụ: Linux distributions như Ubuntu Server, CentOS; Windows Server...).

**Luận ý cơ chế hoạt động DNS:** Như đã giải thích, DNS (Domain Name System) là hệ thống dịch tên miền thành địa chỉ IP. Khi bạn gõ tên miền vào trình duyệt, trình duyệt sẽ hỏi DNS server để tìm địa chỉ IP tương ứng. Sau khi có địa chỉ IP, trình duyệt mới có thể kết nối trực tiếp đến máy chủ web để tải nội dung. DNS là xương sống giúp chúng ta sử dụng tên miền dễ nhớ thay vì phải nhớ các dãy số IP phức tạp.



#### 1.DNS Lookup (Tra cứu DNS)

* **Xuất phát:** Khi bạn nhập một **URL** (ví dụ: https://www.example.com) vào thanh địa chỉ của **Browser** (trình duyệt web của bạn, ví dụ: Chrome, Firefox) và nhấn Enter.
* **Mục đích:** Trình duyệt cần biết địa chỉ IP (Internet Protocol) số của máy chủ chứa website example.com để có thể kết nối với nó. Internet hoạt động dựa trên địa chỉ IP, không phải tên miền dễ nhớ.
* **Cách thức hoạt động:**
  + Trình duyệt kiểm tra bộ nhớ cache DNS cục bộ của nó (đã lưu từ trước đó không?).
  + Nếu không có, nó hỏi hệ điều hành của máy tính.
  + Hệ điều hành hỏi bộ định tuyến (router) của bạn.
  + Router hỏi máy chủ DNS được cấu hình trong mạng của bạn (thường là DNS của nhà cung cấp dịch vụ Internet - ISP của bạn, hoặc các DNS công cộng như Google DNS 8.8.8.8, Cloudflare DNS 1.1.1.1).
  + **DNS server** sẽ tìm kiếm trong cơ sở dữ liệu phân tán khổng lồ của nó để tìm địa chỉ IP tương ứng với tên miền example.com.
  + **Kết quả:** DNS server trả về địa chỉ IP của Origin server (ví dụ: 203.0.113.45) cho trình duyệt. Quá trình này được gọi là "độ phân giải tên miền" (name resolution).

#### 2.TCP Connection (Thiết lập kết nối TCP)

* **Xuất phát:** Sau khi trình duyệt đã có địa chỉ IP của **Origin server**.
* **Mục đích:** Thiết lập một kênh liên lạc ổn định và đáng tin cậy giữa trình duyệt và máy chủ. TCP (Transmission Control Protocol) là giao thức truyền tải chính trên Internet, đảm bảo dữ liệu đến nơi đầy đủ và đúng thứ tự.
* **Cách thức hoạt động (3-way handshake):**
  1. **SYN (Synchronize):** Trình duyệt gửi một gói tin SYN tới Origin server, yêu cầu thiết lập kết nối.
  2. **SYN-ACK (Synchronize-Acknowledge):** Origin server nhận được SYN, gửi lại gói tin SYN-ACK để xác nhận và sẵn sàng kết nối.
  3. **ACK (Acknowledge):** Trình duyệt nhận SYN-ACK, gửi lại gói tin ACK cuối cùng để xác nhận và kết nối TCP được thiết lập thành công.
* **Lưu ý về HTTPS:** Nếu URL là https:// (giao thức an toàn), ngay sau khi kết nối TCP, một quá trình bắt tay (handshake) TLS/SSL sẽ diễn ra để mã hóa kênh liên lạc, đảm bảo dữ liệu truyền đi được bảo mật.

#### 3.HTTP Request (Yêu cầu HTTP)

* **Xuất phát:** Kết nối TCP (hoặc TLS/SSL) đã được thiết lập.
* **Mục đích:** Trình duyệt nói cho Origin server biết nó muốn lấy tài nguyên gì và với cách thức nào.
* **Cách thức hoạt động:** Trình duyệt gửi một thông điệp **HTTP Request** đến Origin server. Thông điệp này bao gồm:
  + **HTTP Method:** (GET, POST, PUT, DELETE, v.v.). Thường là GET để yêu cầu một trang web.
  + **Path:** Đường dẫn đến tài nguyên cụ thể (ví dụ: /, /about-us.html, /images/logo.png).
  + **HTTP Headers:** Các thông tin bổ sung về yêu cầu (ví dụ: User-Agent - loại trình duyệt; Accept - định dạng nội dung chấp nhận; Cookie - dữ liệu phiên).
  + **Body (tùy chọn):** Dữ liệu được gửi lên máy chủ (ví dụ: khi bạn điền và gửi một biểu mẫu dùng phương thức POST).
* **Ví dụ:** GET /index.html HTTP/1.1\r\nHost: www.example.com\r\nUser-Agent: Mozilla/5.0...\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8\r\n\r\n.

#### 4.HTTP Response (Phản hồi HTTP)

* **Xuất phát:** Origin server nhận và xử lý HTTP Request.
* **Mục đích:** Máy chủ gửi lại nội dung mà trình duyệt yêu cầu hoặc một thông báo về kết quả của yêu cầu.
* **Cách thức hoạt động:** Origin server gửi một thông điệp **HTTP Response** về trình duyệt. Thông điệp này bao gồm:
  + **Status Line:** Mã trạng thái (ví dụ: HTTP/1.1 200 OK - thành công; 404 Not Found - không tìm thấy; 500 Internal Server Error - lỗi máy chủ).
  + **HTTP Headers:** Các thông tin bổ sung về phản hồi (ví dụ: Content-Type - loại nội dung, Content-Length - kích thước, Set-Cookie - thiết lập cookie mới).
  + **Body:** Nội dung thực sự của tài nguyên được yêu cầu (ví dụ: mã HTML của trang web, dữ liệu JSON, hình ảnh, file CSS/JS).
* **Ví dụ:** HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nContent-Length: 12345\r\n\r\n<!DOCTYPE html><html><head>...</head><body>...</body></html>

#### 5.Optional Parallel Connections (Các kết nối song song tùy chọn)

* **Mục đích:** Để tải trang web nhanh hơn và hiệu quả hơn.
* **Cách thức hoạt động:**
  + Khi trình duyệt nhận được file HTML đầu tiên từ Origin server, nó sẽ bắt đầu phân tích cú pháp (parse) mã HTML đó.
  + Trong quá trình phân tích, trình duyệt sẽ phát hiện ra các tài nguyên khác cần thiết để hiển thị đầy đủ trang web (ví dụ: hình ảnh, tệp CSS, tệp JavaScript, font chữ, video).
  + Thay vì tải từng tài nguyên một cách tuần tự (đợi cái này xong mới tải cái kia), trình duyệt sẽ mở **nhiều kết nối TCP song song** (thường tối đa 6-8 kết nối cho mỗi tên miền, tùy trình duyệt và phiên bản HTTP) đến Origin server hoặc các máy chủ khác (như Content Delivery Network - CDN) để tải các tài nguyên này cùng một lúc.
  + Điều này giúp rút ngắn đáng kể thời gian tải trang, vì nhiều phần của trang có thể được tải xuống đồng thời.

**1.3 .Web hosting service:**

### ****+Khái niệm về dịch vụ lưu trữ web****

* **Web hosting** là dịch vụ cung cấp **không gian lưu trữ và truy cập trang web** trên Internet.
* Khi bạn tạo website, các tập tin HTML, CSS, hình ảnh, database,... cần được lưu trên **máy chủ (server)** — nơi mà người dùng có thể truy cập thông qua **Internet**.
* Nhà cung cấp dịch vụ web hosting sẽ **cho thuê server** và cung cấp các công cụ quản lý web như FTP, cPanel, email,…

### ****+Khái niệm về** cho thuê domain (tên miền)**

· **Domain (tên miền)** là địa chỉ dễ nhớ của một website (ví dụ: example.com).

· **Cho thuê domain** là dịch vụ đăng ký, giữ quyền sở hữu và sử dụng tên miền trong một khoảng thời gian (thường là 1 năm).

· Bạn có thể thuê tên miền từ các nhà cung cấp như: GoDaddy, Namecheap, Mắt Bão,…

**+ Giải thích các thông số: dung lượng lưu trữ (disk space), băng thông (bandwidth), tên miền  
con (sub-domain), email (có domain riêng), ngôn ngữ lập trình web và cơ sở dữ liệu được web  
server hỗ trợ.**

| **Thông số** | **Giải thích** |
| --- | --- |
| **Disk space (Dung lượng lưu trữ)** | Dung lượng lưu trữ tối đa bạn được phép sử dụng trên server để lưu file web (HTML, hình ảnh, video, database…). |
| **Bandwidth (Băng thông)** | Tổng lưu lượng dữ liệu được truyền tải giữa website và người dùng trong 1 tháng. Tính bằng GB hoặc TB. |
| **Sub-domain (Tên miền con)** | Một phần mở rộng của tên miền chính. Ví dụ: blog.example.com, shop.example.com. Hosting giới hạn số lượng subdomain. |
| **Email (theo tên miền riêng)** | Email có địa chỉ sử dụng tên miền của bạn, ví dụ: admin@example.com. Hosting thường hỗ trợ tạo email này. |
| **Ngôn ngữ lập trình web hỗ trợ** | Cho biết hosting hỗ trợ chạy ngôn ngữ nào như: PHP, ASP.NET, Python, Node.js,... |
| **Cơ sở dữ liệu (Database)** | Hosting có thể hỗ trợ MySQL, MariaDB, MSSQL,… để lưu trữ dữ liệu động cho website như bài viết, người dùng, đơn hàng,... |

**1.4 .Cơ chế hoạt động của client script và server script.**

### 1. Client-Side Scripting (Script phía Client/Trình duyệt)

**Khái niệm:**

* Là các đoạn mã (script) được viết để **chạy trực tiếp trên trình duyệt web của người dùng** (client).
* Trình duyệt nhận mã script này cùng với HTML và CSS, sau đó tự thực thi nó.
* Mục đích chính là để tạo ra các hiệu ứng tương tác, xử lý dữ liệu ngay trên trình duyệt mà không cần phải gửi lại yêu cầu lên máy chủ.

**Ngôn ngữ phổ biến:**

* **JavaScript:** Ngôn ngữ phổ biến nhất và gần như độc quyền cho client-side scripting.
* HTML và CSS không phải là ngôn ngữ script, nhưng chúng là nền tảng để JavaScript tương tác và thay đổi.

**Cơ chế hoạt động:**

1. **Gửi yêu cầu:** Người dùng gửi yêu cầu truy cập một trang web (ví dụ: gõ URL vào thanh địa chỉ).
2. **Máy chủ phản hồi:** Web server gửi lại các file HTML, CSS và **JavaScript** về trình duyệt.
3. **Trình duyệt tải và phân tích:** Trình duyệt tải xuống các file này. Khi gặp các đoạn mã JavaScript (thường nằm trong thẻ <script> hoặc được liên kết từ file .js bên ngoài), nó sẽ:
   * **Phân tích (Parse):** Đọc và hiểu cú pháp của mã JavaScript.
   * **Biên dịch (Compile - JIT):** Các trình duyệt hiện đại có các công cụ biên dịch JIT (Just-In-Time) để chuyển đổi mã JavaScript thành mã máy nhanh hơn.
   * **Thực thi (Execute):** Chạy các lệnh trong mã JavaScript.
4. **Tương tác cục bộ:** Các script này sẽ thực hiện các tác vụ như:
   * **Thay đổi giao diện người dùng (DOM manipulation):** Thay đổi nội dung, màu sắc, vị trí của các phần tử HTML sau khi trang đã tải xong.
   * **Kiểm tra dữ liệu nhập (Form validation):** Kiểm tra xem người dùng đã nhập đúng định dạng thông tin vào các ô trống hay chưa (ví dụ: email đúng định dạng, số điện thoại là số...).
   * **Hiệu ứng động:** Tạo các animation, slideshow, hiệu ứng hover, menu đổ xuống.
   * **Giao tiếp bất đồng bộ (AJAX/Fetch API):** Gửi các yêu cầu nhỏ đến máy chủ để lấy hoặc gửi dữ liệu mà không cần tải lại toàn bộ trang (ví dụ: chức năng "Like" trên Facebook, gợi ý tìm kiếm khi gõ).
   * **Lưu trữ cục bộ:** Sử dụng localStorage hoặc sessionStorage để lưu trữ dữ liệu nhỏ trên trình duyệt.

**Ưu điểm:**

* **Tăng tốc độ phản hồi:** Xử lý ngay trên trình duyệt, không tốn thời gian gửi đi gửi lại máy chủ, giúp website nhanh và mượt mà hơn.
* **Giảm tải cho máy chủ:** Các tác vụ nhỏ được xử lý ở client, giảm bớt gánh nặng cho máy chủ.
* **Giao diện tương tác cao:** Dễ dàng tạo các hiệu ứng, animation phức tạp.

**Nhược điểm:**

* **Phụ thuộc vào trình duyệt:** Mã có thể chạy khác nhau trên các trình duyệt hoặc phiên bản trình duyệt khác nhau.
* **Vấn đề bảo mật:** Mã nguồn dễ dàng bị xem (View Source) và có thể bị lợi dụng nếu không được viết cẩn thận. Không nên xử lý các logic nhạy cảm về bảo mật ở client-side.
* **Vô hiệu hóa Script:** Nếu người dùng tắt JavaScript trong trình duyệt, các chức năng này sẽ không hoạt động.

### 2. Server-Side Scripting (Script phía Máy chủ)

**Khái niệm:**

* Là các đoạn mã (script) được viết để **chạy trên máy chủ web** (server) trước khi nội dung được gửi về trình duyệt của người dùng.
* Mục đích chính là để xử lý logic nghiệp vụ phức tạp, tương tác với cơ sở dữ liệu, quản lý phiên người dùng và tạo ra nội dung động.

**Ngôn ngữ phổ biến:**

* **PHP:** Rất phổ biến cho các website động (WordPress, Laravel).
* **Python:** Với các framework như Django, Flask.
* **Node.js:** (JavaScript chạy trên server)
* **Ruby:** Với framework Ruby on Rails.
* **Java:** Với các framework như Spring, Java EE.
* **.NET (C#):** Từ Microsoft.

**Cơ chế hoạt động:**

1. **Gửi yêu cầu:** Người dùng nhập URL và trình duyệt gửi yêu cầu đến web server.
2. **Máy chủ nhận và xử lý:**
   * Web server (ví dụ: Apache, Nginx) nhận yêu cầu.
   * Nếu yêu cầu trỏ đến một file script server-side (ví dụ: .php, .py, .js cho Node.js), web server sẽ chuyển yêu cầu này cho **bộ xử lý script** tương ứng (ví dụ: PHP interpreter, Python interpreter, Node.js runtime).
   * Bộ xử lý script sẽ **thực thi** đoạn mã server-side. Trong quá trình thực thi này:
     + Nó có thể **truy vấn cơ sở dữ liệu** (Database) để lấy hoặc lưu trữ thông tin (ví dụ: lấy danh sách sản phẩm, lưu thông tin đăng ký của người dùng).
     + Thực hiện các **logic nghiệp vụ** (ví dụ: tính toán giá đơn hàng, xác thực người dùng, xử lý thanh toán).
     + Tạo ra nội dung HTML, CSS, JavaScript (và bất kỳ dữ liệu nào khác) **động** dựa trên dữ liệu từ cơ sở dữ liệu hoặc logic đã xử lý.
3. **Máy chủ phản hồi:** Sau khi script server-side hoàn thành việc tạo ra nội dung, nó gửi nội dung đã tạo (thường là HTML hoàn chỉnh) trở lại web server. Web server sau đó gửi phản hồi HTTP chứa nội dung này về cho trình duyệt của người dùng.
4. **Trình duyệt hiển thị:** Trình duyệt nhận được nội dung HTML đã được tạo sẵn từ máy chủ và hiển thị nó lên màn hình. Lúc này, trình duyệt không cần biết mã nguồn server-side đã chạy như thế nào, nó chỉ nhận được sản phẩm cuối cùng (HTML, CSS, JS client-side).

**Ưu điểm:**

* **Bảo mật cao:** Mã nguồn script phía server không bao giờ được gửi đến trình duyệt, nên không thể bị xem hay giả mạo bởi người dùng. Rất thích hợp để xử lý dữ liệu nhạy cảm, logic kinh doanh quan trọng.
* **Độc lập trình duyệt:** Mã chạy trên máy chủ, nên không phụ thuộc vào loại trình duyệt hay cài đặt của người dùng.
* **Truy cập tài nguyên server:** Có thể tương tác trực tiếp với cơ sở dữ liệu, hệ thống file của server, các dịch vụ khác trên server.
* **Nội dung động:** Tạo ra nội dung tùy chỉnh cho từng người dùng, từng thời điểm.

**Nhược điểm:**

* **Tốn tài nguyên server:** Mỗi yêu cầu đều phải được máy chủ xử lý, có thể gây quá tải nếu lượng truy cập lớn.
* **Thời gian phản hồi chậm hơn:** Mất thời gian để gửi yêu cầu lên server, server xử lý, rồi gửi phản hồi về.
* **Yêu cầu tải lại trang:** Thường yêu cầu tải lại toàn bộ trang (trừ khi kết hợp với AJAX ở client-side).

### Kết luận

Client-side scripting và server-side scripting là hai mặt của một đồng xu trong phát triển web hiện đại. Chúng thường được sử dụng **kết hợp** với nhau để tạo ra các website mạnh mẽ, tương tác và hiệu quả:

* **Server-side** lo các logic phức tạp, bảo mật, truy cập dữ liệu và tạo ra cấu trúc HTML ban đầu.
* **Client-side** lo các tương tác nhỏ, hiệu ứng, kiểm tra dữ liệu cục bộ và cải thiện trải nghiệm người dùng mà không cần gửi mọi thứ lên máy chủ.

Ví dụ, khi bạn điền một biểu mẫu đăng ký:

* **Client-side script** (JavaScript) kiểm tra xem bạn đã điền đủ các trường bắt buộc và định dạng email có đúng không **ngay lập tức** khi bạn nhập.
* Khi bạn nhấn nút "Đăng ký", **server-side script** (ví dụ: PHP) sẽ nhận dữ liệu, kiểm tra lại (để đảm bảo bảo mật), lưu thông tin vào cơ sở dữ liệu, và có thể gửi email xác nhận.

1. **Khái niệm client-script và sử dụng NNLT Javascript**

**Khái niệm Client-Script (Nhắc lại và chi tiết hơn):**

* **Định nghĩa:** Client-script là các đoạn mã lập trình được thiết kế để chạy trực tiếp trên máy tính của người dùng (client), cụ thể là trong trình duyệt web của họ.
* **Mục đích:** Cung cấp khả năng tương tác động, kiểm soát giao diện người dùng, và xử lý dữ liệu cục bộ mà không cần phải gửi yêu cầu trở lại máy chủ cho mỗi thao tác. Điều này giúp giảm tải cho máy chủ, tăng tốc độ phản hồi và cải thiện trải nghiệm người dùng.
* **Cách thức thực thi:** Khi trình duyệt tải một trang web, nó nhận được mã HTML, CSS và các file JavaScript. Trình duyệt có một "engine" (công cụ) JavaScript (ví dụ: V8 của Chrome, SpiderMonkey của Firefox) để đọc, phân tích và thực thi các đoạn mã JavaScript đó.

**2.1 Kiến thức về NNLT thông dịch client-script Javascript.**

**JavaScript (JS) - Ngôn ngữ lập trình thông dịch Client-Script:**

* **Là ngôn ngữ thông dịch (Interpreted Language):**
  + Điều này có nghĩa là mã JavaScript được thực thi "từng dòng" bởi trình duyệt, không cần một bước biên dịch (compilation) riêng biệt thành file thực thi nhị phân (.exe) trước khi chạy như các ngôn ngữ biên dịch (C++, Java).
  + Mặc dù các trình duyệt hiện đại có các công cụ JIT (Just-In-Time) compilation để tối ưu hiệu suất, về cơ bản JavaScript vẫn được coi là ngôn ngữ thông dịch vì quá trình biên dịch diễn ra trong quá trình chạy và không tạo ra file thực thi độc lập.
* **Đa năng:** Ban đầu chỉ dùng cho trình duyệt, nhưng với sự ra đời của Node.js, JavaScript đã mở rộng ra để chạy trên máy chủ (server-side scripting) và thậm chí phát triển ứng dụng di động (React Native) và desktop (Electron). Tuy nhiên, trong ngữ cảnh "client-script", nó vẫn là ngôn ngữ chính cho web browser.
* **Web-centric:** JavaScript được thiết kế để làm việc chặt chẽ với HTML và CSS thông qua **DOM (Document Object Model)**. DOM là một giao diện lập trình ứng dụng (API) cho các tài liệu HTML và XML. JavaScript sử dụng DOM để truy cập và thao tác với các thành phần của trang web, thay đổi nội dung, kiểu dáng, và phản ứng với các sự kiện của người dùng.
* **Bất đồng bộ (Asynchronous):** JavaScript có khả năng thực hiện các tác vụ bất đồng bộ (ví dụ: gửi yêu cầu AJAX đến máy chủ để lấy dữ liệu mà không làm "đơ" trình duyệt). Điều này rất quan trọng để xây dựng các ứng dụng web mượt mà.

**2.2 Kiến thức về sử dụng Javascript. Vd: cú pháp, kiểu dữ liệu, biến, hàm…**

Đây là những kiến thức cơ bản và cốt lõi bạn cần nắm vững khi bắt đầu với JavaScript:

**a. Cú pháp cơ bản:**

* **Cách nhúng JavaScript vào HTML:**
  + Trong thẻ <script>:

<!DOCTYPE html>

<html>

<head>

<title>My Page</title>

</head>

<body>

<h1>Hello World!</h1>

<script>

// Mã JavaScript viết ở đây

alert("Chào mừng bạn đến với trang web!");

</script>

</body>

</html>

* + **File ngoài (.js):** Thường dùng cho mã lớn hơn, giúp tách biệt HTML và JS, dễ quản lý và tái sử dụng.

<!DOCTYPE html>

<html>

<head>

<title>My Page</title>

</head>

<body>

<h1>Hello World!</h1>

<script src="my\_script.js"></script>

</body>

</html>

**b. Kiểu dữ liệu (Data Types):**

JavaScript là ngôn ngữ kiểu động (dynamically typed), nghĩa là bạn không cần khai báo kiểu dữ liệu cho biến, kiểu của biến sẽ được xác định trong quá trình chạy.

* **Primitive Types (Kiểu nguyên thủy):**
  + string**:** Chuỗi ký tự (ví dụ: "Hello", 'World').
  + number**:** Số (bao gồm cả số nguyên và số thực, ví dụ: 10, 3.14).
  + boolean**:** Giá trị logic true hoặc false.
  + undefined**:** Biến đã được khai báo nhưng chưa được gán giá trị.
  + null**:** Giá trị "không có giá trị" một cách có chủ đích.
  + symbol**:** (ES6) Giá trị duy nhất, bất biến.
  + bigint**:** (ES2020) Số nguyên có độ chính xác tùy ý.
* **Non-Primitive Type (Kiểu không nguyên thủy - Object Type):**
  + object**:** Bao gồm các đối tượng thông thường, mảng (array), hàm (function).

**c. Biến (Variables):**

Dùng để lưu trữ dữ liệu.

* var**:** Cách khai báo cũ, có phạm vi hàm (function scope) và có thể bị hoisting (kéo lên đầu phạm vi). **Không khuyến khích sử dụng trong code mới.**

*var message = "Hello";*

* let**:** Khai báo biến có thể thay đổi giá trị. Có phạm vi khối (block scope). **Nên dùng.**

*let count = 0;*

*count = 1; // Có thể thay đổi giá trị*

* const**:** Khai báo hằng số (không thể gán lại giá trị sau khi khai báo). Có phạm vi khối (block scope). **Nên dùng cho các giá trị không đổi.**

const PI = 3.14;

**d. Toán tử (Operators):**

* **Số học:** +, -, \*, /, % (chia lấy dư), \*\* (lũy thừa).
* **Gán:** =, +=, -=, \*=, /=.
* **So sánh:** == (so sánh giá trị), === (so sánh giá trị và kiểu), !=, !==, >, <, >=, <=.
* **Logic:** && (AND), || (OR), ! (NOT).

**e. Cấu trúc điều khiển (Control Structures):**

* **Câu lệnh điều kiện:** if, else if, else

*let age = 20;if (age >= 18) {*

*console.log("Đủ tuổi bầu cử.");*

*} else {*

*console.log("Chưa đủ tuổi.");*

*}*

* **Vòng lặp:** for, while, do...while, for...in, for...of

*for (let i = 0; i < 5; i++) {*

*console.log(i); // In ra 0, 1, 2, 3, 4*

*}*

* **switch:**

*let day = "Monday";switch (day) {*

*case "Monday":*

*console.log("It's Monday!");*

*break;*

*case "Friday":*

*console.log("It's Friday!");*

*break;*

*default:*

*console.log("Some other day.");*

*}*

**f. Hàm (Functions):**

Là các khối mã có thể tái sử dụng để thực hiện một tác vụ cụ thể.

* **Khai báo hàm:**

*function greet(name) {*

*return "Hello, " + name + "!";*

*}let message = greet("Alice"); // Gọi hàmconsole.log(message); // Output: Hello, Alice!*

* **Hàm biểu thức (Function Expression):**

*const square = function(num) {*

*return num \* num;*

*};console.log(square(5)); // Output: 25*

* **Hàm mũi tên (Arrow Function - ES6):** Cú pháp ngắn gọn hơn, thường dùng cho hàm callback.

*const add = (a, b) => a + b;*

*console.log(add(2, 3)); // Output: 5*

**g. Đối tượng (Objects):**

Trong JavaScript, hầu hết mọi thứ đều là đối tượng hoặc có thể được coi là đối tượng. Đối tượng là tập hợp các cặp khóa-giá trị (key-value pairs).

*let person = {*

*firstName: "John",*

*lastName: "Doe",*

*age: 30,*

*greet: function() {*

*console.log("Hi, " + this.firstName);*

*}*

*};*

*console.log(person.firstName); // Output: John*

*person.greet(); // Output: Hi, John*

**h. Mảng (Arrays):**

Là một loại đối tượng dùng để lưu trữ danh sách các giá trị.

JavaScript

let fruits = ["apple", "banana", "cherry"];console.log(fruits[0]); // Output: apple

fruits.push("orange"); // Thêm phần tử vào cuốiconsole.log(fruits.length); // Output: 4

**2.3 Khái niệm cơ bản về XSS. Từ khóa: “cross-site scripting”**

**Khái niệm:**

* **XSS (Cross-Site Scripting)** là một loại lỗ hổng bảo mật phổ biến trong các ứng dụng web.
* Nó xảy ra khi một kẻ tấn công có thể **chèn mã độc (thường là JavaScript)** vào một trang web hợp pháp, và mã đó được thực thi trên trình duyệt của người dùng khác truy cập trang web đó.
* Bởi vì mã độc chạy trong ngữ cảnh của trang web hợp pháp, nó có thể truy cập cookie, session token, hoặc các thông tin nhạy cảm khác mà trình duyệt giữ cho trang web đó.
* Mục tiêu chính của XSS là đánh cắp thông tin người dùng, chiếm quyền điều khiển phiên, hoặc chuyển hướng người dùng đến các trang web độc hại.

**Cơ chế hoạt động (Tổng quát):**

1. **Kẻ tấn công chèn mã:** Kẻ tấn công tìm cách đưa mã JavaScript độc hại vào một trang web dễ bị tấn công. Điều này thường xảy ra thông qua các trường nhập liệu không được kiểm tra kỹ lưỡng (ví dụ: bình luận, diễn đàn, thanh tìm kiếm).
   * **Ví dụ:** Thay vì nhập tên, kẻ tấn công nhập <script>alert('You are hacked!');</script>.
2. **Máy chủ lưu trữ (hoặc phản ánh):** Máy chủ web lưu trữ (Persistent XSS) hoặc phản ánh (Reflected XSS) nội dung có chứa mã độc này mà không loại bỏ nó.
3. **Nạn nhân truy cập:** Một người dùng (nạn nhân) truy cập trang web bị tấn công.
4. **Trình duyệt thực thi mã độc:** Trình duyệt của nạn nhân tải trang web. Khi gặp mã JavaScript độc hại, trình duyệt sẽ coi nó là một phần hợp lệ của trang web và **thực thi nó**.
5. **Tấn công xảy ra:** Mã JavaScript độc hại có thể:
   * Đánh cắp cookie của nạn nhân (có thể chứa thông tin phiên đăng nhập) và gửi về cho kẻ tấn công.
   * Chuyển hướng nạn nhân đến một trang web lừa đảo.
   * Thay đổi nội dung hiển thị của trang web để lừa đảo người dùng.
   * Thực hiện các yêu cầu HTTP dưới danh nghĩa của nạn nhân (ví dụ: thay đổi mật khẩu).

**Các loại XSS phổ biến:**

* **Stored/Persistent XSS:** Mã độc được lưu trữ vĩnh viễn trên máy chủ (ví dụ: trong cơ sở dữ liệu) và được phục vụ cho mọi người dùng truy cập trang đó. Đây là loại nguy hiểm nhất.
* **Reflected XSS:** Mã độc được phản ánh ngay lập tức từ yêu cầu của người dùng lên trang web. Ví dụ: khi bạn tìm kiếm ">&lt;script>alert('XSS');&lt;/script>" trên một trang web không bảo mật, và trang đó hiển thị lại chính chuỗi tìm kiếm của bạn mà không xử lý.
* **DOM-based XSS:** Lỗ hổng nằm ở cách mã JavaScript phía client xử lý dữ liệu đầu vào mà không thông qua máy chủ.

**Cách phòng tránh cơ bản:**

* **Input Validation (Kiểm tra đầu vào):** Luôn luôn kiểm tra và làm sạch (sanitize) mọi dữ liệu đầu vào từ người dùng trên cả client-side và đặc biệt là server-side.
* **Output Encoding/Escaping (Mã hóa/Thoát đầu ra):** Trước khi hiển thị dữ liệu do người dùng cung cấp lên trang web, hãy mã hóa hoặc thoát các ký tự đặc biệt (như <, >, ", ', &) để chúng được hiển thị dưới dạng văn bản thuần túy thay vì được trình duyệt hiểu là mã HTML hoặc JavaScript.

Việc hiểu XSS là cực kỳ quan trọng đối với bất kỳ ai làm việc với phát triển web, đặc biệt là client-side scripting, để xây dựng các ứng dụng an toàn.

**3. Khái niệm & sử dụng mô hình Javascript + HTML DOM, xử lý sự kiện.**

3.1 Kiến thức về sự kiện và xử lý sự kiện.

**a. Khái niệm về HTML DOM (Document Object Model)**

* HTML DOM là một **giao diện lập trình ứng dụng (API)** cho các tài liệu HTML. Nó định nghĩa cách các thành phần HTML được biểu diễn dưới dạng các đối tượng (objects) và cách JavaScript có thể truy cập và thay đổi các đối tượng đó.

Hãy tưởng tượng trang web của bạn như một cây gia phả:

* Toàn bộ tài liệu HTML là **root** (gốc).
* Các thẻ như <html>, <head>, <body> là các **node** (nút).
* Mỗi thẻ HTML (ví dụ: <p>, <h1>, <div>, <input>) là một **element node**.
* Nội dung văn bản bên trong các thẻ là **text node**.
* Các thuộc tính của thẻ (ví dụ: id, class, src) là **attribute node**.

· **Mục đích của DOM:** Cung cấp một cấu trúc chuẩn (cây đối tượng) để JavaScript có thể:

* **Truy cập:** Tìm kiếm các phần tử HTML trên trang.
* **Thay đổi:** Thay đổi nội dung văn bản, thuộc tính, hoặc kiểu dáng (CSS) của các phần tử.
* **Thêm/Xóa:** Thêm các phần tử HTML mới vào trang hoặc xóa các phần tử hiện có.
* **Xử lý sự kiện:** Phản ứng với các tương tác của người dùng.

· **Cách JavaScript tương tác với DOM:**

· JavaScript sử dụng các phương thức và thuộc tính của đối tượng document để truy cập các phần tử HTML.

* **Các phương thức truy cập phổ biến:**
  + document.getElementById('id-của-phần-tử'): Truy cập một phần tử duy nhất bằng ID.
  + document.getElementsByClassName('class-của-phần-tử'): Truy cập một tập hợp các phần tử bằng tên lớp (trả về HTMLCollection).
  + document.getElementsByTagName('tên-thẻ'): Truy cập một tập hợp các phần tử bằng tên thẻ (trả về HTMLCollection).
  + document.querySelector('selector-css'): Truy cập phần tử đầu tiên khớp với CSS selector.
  + document.querySelectorAll('selector-css'): Truy cập tất cả các phần tử khớp với CSS selector (trả về NodeList).

**b. Khái niệm về Sự kiện (Events):**

* **Sự kiện là gì?** Trong lập trình web, sự kiện là những gì xảy ra trong trình duyệt mà JavaScript có thể "nghe" và phản ứng lại. Đó là các tín hiệu cho biết một điều gì đó đã xảy ra trên trang web.
* **Ví dụ về các sự kiện thông dụng:**
  1. **Click chuột:** click (khi người dùng nhấp vào một phần tử).
  2. **Di chuột qua/rời khỏi:** mouseover, mouseout.
  3. **Nhập liệu bàn phím:** keydown, keyup, keypress.
  4. **Thay đổi giá trị:** change (khi giá trị của input, select, textarea thay đổi).
  5. **Gửi biểu mẫu:** submit (khi một form được gửi đi).
  6. **Tải trang:** DOMContentLoaded (khi DOM đã được tải hoàn chỉnh), load (khi toàn bộ trang, bao gồm hình ảnh, đã tải xong).
  7. **Tải lỗi:** error.

**c. Khái niệm về Xử lý Sự kiện (Event Handling):**

* **Xử lý sự kiện là gì?** Là quá trình JavaScript phản ứng lại khi một sự kiện xảy ra. Nó liên quan đến việc "gán" một hàm (gọi là **event handler** hoặc **event listener**) vào một phần tử cụ thể, để khi sự kiện đó xảy ra trên phần tử đó, hàm này sẽ được thực thi.
* Các cách gán trình xử lý sự kiện:

1.**Sử dụng thuộc tính HTML (Inline Event Handlers - Không khuyến khích):**

* Trực tiếp viết mã JavaScript vào thuộc tính HTML của thẻ.
* Cách dùng: onclick="myFunction()"
* Nhược điểm: Khó bảo trì, trộn lẫn HTML và JS, không linh hoạt.

<button onclick="alert('Button clicked inline!')">Click Me</button>

2.**Sử dụng thuộc tính DOM (Traditional DOM Event Handlers - Đã cũ, có thể bị ghi đè):**

* Gán một hàm cho thuộc tính sự kiện của đối tượng DOM.
* Cách dùng: element.onclick = functionName;
* Nhược điểm: Một sự kiện chỉ có thể có một trình xử lý duy nhất. Nếu gán lại, cái cũ sẽ bị ghi đè.
* Ví dụ:

const myButton = document.getElementById('myBtn');

myButton.onclick = function() {

console.log('Button clicked using .onclick!');

};

3.**Sử dụng** addEventListener() **(Khuyến khích):**

* Phương pháp hiện đại và linh hoạt nhất. Cho phép thêm nhiều trình xử lý cho cùng một sự kiện trên cùng một phần tử mà không bị ghi đè.
* Cách dùng: element.addEventListener('event-type', functionName, [useCapture]);
* event-type: Tên sự kiện (không có "on" phía trước, ví dụ: 'click' thay vì 'onclick').
* functionName: Hàm sẽ được gọi khi sự kiện xảy ra.
* useCapture (tùy chọn): Boolean, true cho giai đoạn "capturing", false (mặc định) cho giai đoạn "bubbling". (Khái niệm Event Bubbling/Capturing).
* VÍ Dụ:

const myButton = document.getElementById('myBtn');

myButton.addEventListener('click', function() {

console.log('Button clicked using addEventListener!');

});

// Có thể thêm một handler khác cho cùng sự kiện

myButton.addEventListener('click', function() {

alert('Another click handler!');

});

3.2 Ứng dụng xử lý một số sự kiện thông dụng (đã giới thiệu trên lớp).

Dưới đây là một số ví dụ minh họa cách sử dụng JavaScript và DOM để xử lý các sự kiện phổ biến:

**Ví dụ 1: Thay đổi nội dung khi click nút**

HTML

<!DOCTYPE html><html><head>

<title>Event Example</title></head><body>

<h1 id="myHeading">Chào mừng!</h1>

<button id="changeTextBtn">Đổi chữ</button>

<script>

// 1. Lấy tham chiếu đến phần tử HTML

const headingElement = document.getElementById('myHeading');

const buttonElement = document.getElementById('changeTextBtn');

// 2. Định nghĩa hàm xử lý sự kiện

function handleChangeText() {

headingElement.textContent = "Bạn vừa đổi chữ thành công!"; // Thay đổi nội dung văn bản

headingElement.style.color = "blue"; // Thay đổi màu chữ

}

// 3. Gán hàm xử lý sự kiện cho sự kiện 'click' của nút

buttonElement.addEventListener('click', handleChangeText);

</script></body></html>

**Ví dụ 2: Kiểm tra dữ liệu khi nhập liệu (Event** input **hoặc** change**)**

Giả sử bạn muốn kiểm tra xem người dùng có đang nhập số vào một ô input hay không.

HTML

<!DOCTYPE html><html><head>

<title>Input Validation</title></head><body>

<label for="numberInput">Nhập số:</label>

<input type="text" id="numberInput">

<p id="message" style="color: red;"></p>

<script>

const numberInput = document.getElementById('numberInput');

const messageParagraph = document.getElementById('message');

numberInput.addEventListener('input', function() {

const inputValue = this.value; // Lấy giá trị hiện tại của input

if (isNaN(inputValue) && inputValue !== "") { // isNaN kiểm tra "có phải là Not-a-Number"

messageParagraph.textContent = "Chỉ được nhập số!";

} else {

messageParagraph.textContent = ""; // Xóa thông báo lỗi

}

});

</script></body></html>

**Ví dụ 3: Xử lý sự kiện gửi form (**submit**)**

Ngăn chặn việc form gửi đi mặc định và thực hiện validation trước.

HTML

<!DOCTYPE html><html><head>

<title>Form Submit Example</title></head><body>

<form id="myForm">

<label for="username">Tên người dùng:</label>

<input type="text" id="username" required>

<br>

<label for="password">Mật khẩu:</label>

<input type="password" id="password" required>

<br>

<button type="submit">Đăng nhập</button>

</form>

<p id="loginMessage" style="color: green;"></p>

<script>

const myForm = document.getElementById('myForm');

const usernameInput = document.getElementById('username');

const passwordInput = document.getElementById('password');

const loginMessage = document.getElementById('loginMessage');

myForm.addEventListener('submit', function(event) {

event.preventDefault(); // NGĂN CHẶN HÀNH VI MẶC ĐỊNH CỦA FORM

const username = usernameInput.value;

const password = passwordInput.value;

if (username === "" || password === "") {

loginMessage.textContent = "Vui lòng điền đầy đủ tên người dùng và mật khẩu!";

loginMessage.style.color = "red";

} else if (username === "admin" && password === "123") {

loginMessage.textContent = "Đăng nhập thành công!";

loginMessage.style.color = "green";

// Ở đây bạn có thể chuyển hướng trang hoặc thực hiện các hành động khác

} else {

loginMessage.textContent = "Tên người dùng hoặc mật khẩu không đúng!";

loginMessage.style.color = "red";

}

});

</script></body></html>

3.3 Sử dụng javascript có ứng dụng regex để kiểm tra định dạng khi nhập liệu.

**Khái niệm Regex (Biểu thức chính quy):**

* **Regex** là một chuỗi ký tự đặc biệt được sử dụng để **tìm kiếm và khớp các mẫu (patterns)** trong văn bản. Nó là một công cụ cực kỳ mạnh mẽ để kiểm tra định dạng của chuỗi.
* Trong JavaScript, Regex là một đối tượng (RegExp) và có các phương thức để làm việc với chuỗi.

**Cú pháp Regex cơ bản:**

* Regex có thể được tạo bằng cách sử dụng literal (/pattern/flags) hoặc constructor (new RegExp('pattern', 'flags')).
* **Các ký tự đặc biệt thường dùng:**
  + .: Bất kỳ ký tự nào (trừ xuống dòng).
  + \d: Một chữ số (0-9).
  + \w: Một ký tự chữ, số hoặc dấu gạch dưới (word character).
  + \s: Một ký tự khoảng trắng (space character).
  + +: Một hoặc nhiều lần xuất hiện của ký tự/nhóm trước đó.
  + \*: Không hoặc nhiều lần xuất hiện.
  + ?: Không hoặc một lần xuất hiện.
  + {n}: Xuất hiện chính xác n lần.
  + {n,m}: Xuất hiện từ n đến m lần.
  + {n,}: Xuất hiện ít nhất n lần.
  + ^: Bắt đầu chuỗi.
  + $: Kết thúc chuỗi.
  + [abc]: Một trong các ký tự a, b, c.
  + [a-z]: Bất kỳ ký tự chữ cái thường từ a đến z.
  + [^abc]: Bất kỳ ký tự nào KHÔNG phải a, b, c.
  + |: Hoặc.
  + ( ): Nhóm các ký tự.

**Ứng dụng Regex trong kiểm tra định dạng (Validation):**

**Ví dụ: Kiểm tra định dạng Email**

HTML

<!DOCTYPE html><html><head>

<title>Email Validation with Regex</title></head><body>

<label for="emailInput">Nhập Email:</label>

<input type="text" id="emailInput">

<p id="emailMessage" style="color: red;"></p>

<script>

const emailInput = document.getElementById('emailInput');

const emailMessage = document.getElementById('emailMessage');

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

emailInput.addEventListener('input', function() {

const emailValue = this.value;

if (emailValue === "") {

emailMessage.textContent = "Email không được để trống.";

emailMessage.style.color = "red";

} else if (!emailRegex.test(emailValue)) { // .test() là phương thức của Regex để kiểm tra khớp mẫu

emailMessage.textContent = "Định dạng email không hợp lệ.";

emailMessage.style.color = "red";

} else {

emailMessage.textContent = "Định dạng email hợp lệ.";

emailMessage.style.color = "green";

}

});

</script>

</body>

</html>

**4. Khái niệm server-script và sử dụng NNLT PHP**4.1 Kiến thức về NNLT thông dịch server-script PHP

**Khái niệm Server-Script (Nhắc lại và chi tiết hơn):**

* **Server-Script** là các đoạn mã chương trình được thiết kế để **thực thi trên máy chủ web** (server) trước khi nội dung được gửi về trình duyệt của người dùng (client).
* **Mục đích chính:**
  + **Xử lý logic nghiệp vụ phức tạp:** Các tác vụ như xác thực người dùng, xử lý đơn hàng, tính toán phức tạp.
  + **Tương tác với cơ sở dữ liệu (Database):** Lưu trữ, truy xuất, cập nhật, xóa dữ liệu (ví dụ: thông tin người dùng, bài viết, sản phẩm).
  + **Tạo nội dung động:** Sinh ra các trang HTML tùy chỉnh dựa trên dữ liệu từ database hoặc tương tác của người dùng.
  + **Quản lý phiên (Session) và cookie:** Duy trì trạng thái người dùng giữa các lần truy cập trang.
  + **Bảo mật:** Xử lý các logic nhạy cảm về bảo mật,

**PHP - Ngôn ngữ lập trình thông dịch Server-Script:**

* **Là ngôn ngữ thông dịch (Interpreted Language):** Tương tự như JavaScript ở client-side, mã PHP được thực thi (thông dịch) bởi một **PHP Interpreter** (thường là PHP-FPM hoặc module của Apache/Nginx) trên máy chủ mà không cần biên dịch thành file thực thi độc lập trước đó.
* **Mở nguồn (Open Source) và Miễn phí:** Điều này làm cho PHP rất phổ biến và có cộng đồng lớn.
* **Nhúng vào HTML:** PHP được thiết kế để dễ dàng nhúng trực tiếp vào các tài liệu HTML.
* **Đa nền tảng:** Chạy được trên nhiều hệ điều hành khác nhau (Windows, Linux, macOS) và với nhiều máy chủ web (Apache, Nginx, IIS).
* **Tích hợp mạnh mẽ với Database:** Hỗ trợ tốt cho nhiều hệ quản trị cơ sở dữ liệu phổ biến như MySQL, PostgreSQL, SQL Server.

4.2 Phân biệt cơ chế hoạt động client-script và server-script

Đây là điểm cốt lõi để hiểu rõ sự khác biệt giữa hai loại script này:

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **Client-Script (Ví dụ: JavaScript)** | **Server-Script (Ví dụ: PHP)** |
| **Nơi thực thi** | Trình duyệt web của người dùng (máy client). | Máy chủ web (server). |
| **Khi nào chạy** | Sau khi trang web được tải về trình duyệt. | Trước khi trang web được gửi đi từ máy chủ. |
| **Mã nguồn** | **Có thể xem được** bằng cách "View Source" trong trình duyệt. | **Không thể xem được** từ trình duyệt (chỉ output HTML cuối cùng). |
| **Truy cập tài nguyên** | Chỉ giới hạn trong môi trường trình duyệt (DOM, LocalStorage). | Có thể truy cập file hệ thống, cơ sở dữ liệu, tài nguyên mạng trên server. |
| **Mục đích** | Tương tác giao diện người dùng, hiệu ứng động, kiểm tra dữ liệu sơ bộ. | Xử lý logic nghiệp vụ, quản lý dữ liệu, bảo mật, tạo nội dung động. |
| **Tốc độ phản hồi** | Nhanh, vì xử lý ngay trên máy người dùng, không cần qua mạng. | Chậm hơn (phải gửi yêu cầu lên server, server xử lý rồi gửi về). |
| **Bảo mật** | Thấp (mã nguồn dễ lộ, dễ bị giả mạo). | Cao (mã nguồn được giữ bí mật trên server). |
| **Phụ thuộc** | Vào trình duyệt và phiên bản trình duyệt. | Vào máy chủ web và phiên bản ngôn ngữ script trên server. |
| **Giao tiếp** | Với máy chủ thông qua AJAX/Fetch API (nếu cần dữ liệu động). | Giao tiếp với Database, các dịch vụ backend khác trên server. |
| **Kết quả trả về** | Không có kết quả trả về cho máy chủ (trừ khi dùng AJAX). | Trả về một trang HTML, CSS, JavaScript (có thể đã được tạo động) cho trình duyệt. |

4.3 Kiến thức về sử dụng PHP. Vd: cú pháp, kiểu dữ liệu, biến, hàm…

**a. Cú pháp cơ bản:**

* **Dấu mở/đóng PHP:** Mã PHP phải nằm trong các cặp thẻ <?php và ?>.

Ví dụ:

<?php

// Mã PHP của bạn ở đây

echo "Hello, World!";

?>

**b. Biến (Variables):**

· Bắt đầu bằng dấu $.

· PHP là ngôn ngữ kiểu động, không cần khai báo kiểu dữ liệu.

Ví du:

$name = "Alice"; // string

$age = 30; // integer

$price = 19.99; // float

$isStudent = true; // boolean

**c. Kiểu dữ liệu (Data Types):**

* Scalar types(kiểu vô hướng)
  + String:chuỗi ký tự
  + Interger : số nguyên
  + Float : số thực
  + Boolean: true
* Compound types(kiểu phức hợp)
  + Array: Mảng
  + Object:L Đối tượng

**d. Toán tử (Operators):**

* Số học : +, -, \*, /, % (chia lấy dư), \*\* (lũy thừa).
* Gán: =, +=, -=, \*=, /=.
* So sánh : == , === , !=, !==, >, <, >=, <=.
* Logic : && (AND), || (OR), ! (NOT), and, or, xor.
* Nối chuỗi : .

**e. Hàm (Functions):**

* Là các khối mã có thể tái sử dụng.

Ví dụ:

function greet($name) {

return "Hello, " . $name . "!";

}

echo greet("Bob"); // Output: Hello, Bob!  
4.4. Mục đích và cú pháp PHP

· **Mục đích chính của PHP:**

· **Tạo ra các trang web động (Dynamic Web Pages):** Nội dung trang web có thể thay đổi dựa trên dữ liệu từ cơ sở dữ liệu, tương tác của người dùng, hoặc các điều kiện khác.

* **Thu thập dữ liệu từ biểu mẫu HTML (Form Data Handling):** Xử lý dữ liệu do người dùng gửi qua các biểu mẫu (POST/GET).
* **Tương tác với cơ sở dữ liệu:** Kết nối, truy vấn, thêm, sửa, xóa dữ liệu trong các hệ quản trị CSDL như MySQL.
* **Quản lý phiên (Session Management):** Duy trì trạng thái người dùng giữa các lần truy cập.
* **Tạo các API (Application Programming Interface):** Cung cấp dữ liệu cho các ứng dụng khác (ví dụ: ứng dụng di động, ứng dụng JavaScript).
* **Xử lý file:** Đọc, ghi, xóa file trên server.
* **Gửi email.**
* **Kiểm soát truy cập người dùng.**

· **Cú pháp PHP (nhắc lại):**

* Mã PHP luôn được đặt trong cặp thẻ mở và đóng: <?php ... ?>.
* Mỗi câu lệnh kết thúc bằng dấu chấm phẩy ;.
* Biến bắt đầu bằng $.
* Hàm và các cấu trúc điều khiển tương tự C/C++/Java về mặt cú pháp { } và ( ).

4.5. PHP cơ bản và xử lý nhập xuất dữ liệu trên web

**a. Xuất dữ liệu ra trình duyệt:**

* echo**:** Dùng để xuất chuỗi, số, biến ra trình duyệt. Không có giá trị trả về.

<?php

echo "<h1>Welcome to my PHP page!</h1>";

$productName = "Laptop";

echo "<p>Product: " . $productName . "</p>";

?>

**b. Nhập dữ liệu từ người dùng (Form Data):**

PHP xử lý dữ liệu gửi từ biểu mẫu HTML bằng cách sử dụng các biến siêu toàn cục (superglobals): $\_GET và $\_POST.

* $\_GET:
  + Dùng để thu thập dữ liệu từ form có method = "get".
  + Dữ liệu được gửi qua URL dưới dạng các cặp key-value (query string).
  + Thích hợp cho các yêu cầu lấy dữ liệu (ví dụ: tìm kiếm, phân trang) vì dễ dàng bookmark hoặc chia sẻ URL.
  + **Không an toàn cho dữ liệu nhạy cảm** vì dữ liệu hiển thị trên URL.
  + **Giới hạn kích thước dữ liệu.**

4.6. PHP session và Cookie.

Đây là hai cơ chế quan trọng để duy trì trạng thái của người dùng (state management) trong môi trường web phi trạng thái (stateless).

**a. Cookie:**

* **Khái niệm:** Là các file văn bản nhỏ được **lưu trữ trên trình duyệt của người dùng** bởi máy chủ web.
* **Mục đích:**
  1. Lưu trữ các cài đặt của người dùng (ví dụ: ngôn ngữ, chủ đề).
  2. Ghi nhớ thông tin đăng nhập (chức năng "ghi nhớ tôi").
  3. Theo dõi hành vi người dùng (tracking).
* **Cơ chế hoạt động:**
  1. Máy chủ gửi một cookie đến trình duyệt thông qua header HTTP Set-Cookie.
  2. Trình duyệt lưu trữ cookie đó.
  3. Mỗi lần trình duyệt gửi yêu cầu đến cùng một máy chủ (cùng tên miền), nó sẽ tự động gửi lại tất cả các cookie liên quan trong header HTTP Cookie.
  4. Máy chủ nhận lại cookie và sử dụng thông tin trong đó.
* **Hàm trong PHP:**
  1. setcookie(name, value, expire, path, domain, secure, httponly): Dùng để tạo hoặc cập nhật một cookie. Phải được gọi trước khi bất kỳ output nào được gửi đến trình duyệt.
  2. $\_COOKIE: Biến siêu toàn cục dùng để truy cập các cookie đã gửi từ trình duyệt.
* Ví du:

<?php

// Đặt cookie tên "username" với giá trị "JohnDoe", hết hạn sau 1 giờ

setcookie("username", "JohnDoe", time() + 3600, "/");

if (isset($\_COOKIE['username'])) {

echo "Welcome back, " . htmlspecialchars($\_COOKIE['username']) . "!";

} else {

echo "Hello, guest!";

}

?>

**b. Session (Phiên làm việc):**

* **Khái niệm:** Là một cách để **lưu trữ thông tin người dùng trên máy chủ** trong suốt một phiên làm việc (session). Thông tin này không được lưu trên trình duyệt của người dùng, mà được liên kết với một ID phiên duy nhất.
* **Mục đích:** Duy trì trạng thái đăng nhập, giỏ hàng, thông tin tạm thời khác của người dùng trong suốt quá trình họ duyệt web.
* **Cơ chế hoạt động:**
  1. Khi người dùng truy cập trang web lần đầu (hoặc khi bắt đầu một phiên), máy chủ tạo ra một **ID phiên duy nhất** và lưu trữ nó.
  2. ID phiên này được gửi đến trình duyệt dưới dạng một **cookie đặc biệt** (thường tên là PHPSESSID).
  3. Trình duyệt lưu cookie ID phiên này.
  4. Mỗi lần người dùng gửi yêu cầu tiếp theo, trình duyệt gửi lại cookie ID phiên.
  5. Máy chủ nhận ID phiên, dùng nó để truy xuất các dữ liệu đã lưu trữ cho phiên đó.
* **Hàm trong PHP:**
  1. session\_start(): **Bắt buộc phải gọi ở đầu mỗi trang PHP** (trước bất kỳ output nào) để bắt đầu hoặc tiếp tục một phiên.
  2. $\_SESSION: Biến siêu toàn cục dạng mảng, dùng để lưu trữ và truy xuất dữ liệu phiên.
  3. session\_destroy(): Hủy bỏ toàn bộ phiên (thường dùng khi đăng xuất).
  4. unset($\_SESSION['key']): Xóa một biến cụ thể trong phiên.
* Ví dụ:

<?php

session\_start(); // Bắt đầu phiên làm việc

if (isset($\_SESSION['views'])) {

$\_SESSION['views'] = $\_SESSION['views'] + 1;

} else {

$\_SESSION['views'] = 1;

}

echo "Bạn đã xem trang này " . $\_SESSION['views'] . " lần trong phiên này.";

// Đăng nhập

if (isset($\_POST['login'])) {

// Giả sử xác thực thành công

$\_SESSION['user\_id'] = 123;

$\_SESSION['username'] = "admin";

echo "<p>Bạn đã đăng nhập thành công!</p>";

}

// Đăng xuất

if (isset($\_POST['logout'])) {

session\_unset(); // Xóa tất cả biến session

session\_destroy(); // Hủy session

echo "<p>Bạn đã đăng xuất.</p>";

}

?>

<form method="post">

<input type="submit" name="login" value="Đăng nhập">

<input type="submit" name="logout" value="Đăng xuất">

</form>

**Ưu điểm của Session so với Cookie:**

* **Bảo mật hơn:** Dữ liệu thực sự được lưu trên server, không bị lộ ra client.
* **Không giới hạn kích thước:** Có thể lưu trữ lượng lớn dữ liệu.
* **Dễ dàng kiểm soát thời gian sống.**

4.7. Lập trình hướng đối tượng trong PHP

Được rồi, chúng ta sẽ đi sâu vào phần Server-Side Scripting với PHP, bao gồm khái niệm, cách hoạt động, kiến thức cơ bản và các phần quan trọng khác như Session, Cookie, và Lập trình hướng đối tượng.

### 4. Khái niệm Server-Script và Sử dụng NNLT PHP

#### 4.1 Kiến thức về NNLT thông dịch Server-Script PHP

* **Khái niệm Server-Script (Nhắc lại và chi tiết hơn):**
  + **Server-Script** là các đoạn mã chương trình được thiết kế để **thực thi trên máy chủ web** (server) trước khi nội dung được gửi về trình duyệt của người dùng (client).
  + **Mục đích chính:**
    - **Xử lý logic nghiệp vụ phức tạp:** Các tác vụ như xác thực người dùng, xử lý đơn hàng, tính toán phức tạp.
    - **Tương tác với cơ sở dữ liệu (Database):** Lưu trữ, truy xuất, cập nhật, xóa dữ liệu (ví dụ: thông tin người dùng, bài viết, sản phẩm).
    - **Tạo nội dung động:** Sinh ra các trang HTML tùy chỉnh dựa trên dữ liệu từ database hoặc tương tác của người dùng.
    - **Quản lý phiên (Session) và cookie:** Duy trì trạng thái người dùng giữa các lần truy cập trang.
    - **Bảo mật:** Xử lý các logic nhạy cảm về bảo mật, vì mã nguồn không bao giờ được gửi đến trình duyệt.
* **PHP - Ngôn ngữ lập trình thông dịch Server-Script:**
  + **Là ngôn ngữ thông dịch (Interpreted Language):** Tương tự như JavaScript ở client-side, mã PHP được thực thi (thông dịch) bởi một **PHP Interpreter** (thường là PHP-FPM hoặc module của Apache/Nginx) trên máy chủ mà không cần biên dịch thành file thực thi độc lập trước đó.
  + **Mở nguồn (Open Source) và Miễn phí:** Điều này làm cho PHP rất phổ biến và có cộng đồng lớn.
  + **Nhúng vào HTML:** PHP được thiết kế để dễ dàng nhúng trực tiếp vào các tài liệu HTML.
  + **Đa nền tảng:** Chạy được trên nhiều hệ điều hành khác nhau (Windows, Linux, macOS) và với nhiều máy chủ web (Apache, Nginx, IIS).
  + **Tích hợp mạnh mẽ với Database:** Hỗ trợ tốt cho nhiều hệ quản trị cơ sở dữ liệu phổ biến như MySQL, PostgreSQL, SQL Server.

#### 4.2 Phân biệt cơ chế hoạt động Client-Script và Server-Script

Đây là điểm cốt lõi để hiểu rõ sự khác biệt giữa hai loại script này:

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **Client-Script (Ví dụ: JavaScript)** | **Server-Script (Ví dụ: PHP)** |
| **Nơi thực thi** | Trình duyệt web của người dùng (máy client). | Máy chủ web (server). |
| **Khi nào chạy** | Sau khi trang web được tải về trình duyệt. | Trước khi trang web được gửi đi từ máy chủ. |
| **Mã nguồn** | **Có thể xem được** bằng cách "View Source" trong trình duyệt. | **Không thể xem được** từ trình duyệt (chỉ output HTML cuối cùng). |
| **Truy cập tài nguyên** | Chỉ giới hạn trong môi trường trình duyệt (DOM, LocalStorage). | Có thể truy cập file hệ thống, cơ sở dữ liệu, tài nguyên mạng trên server. |
| **Mục đích** | Tương tác giao diện người dùng, hiệu ứng động, kiểm tra dữ liệu sơ bộ. | Xử lý logic nghiệp vụ, quản lý dữ liệu, bảo mật, tạo nội dung động. |
| **Tốc độ phản hồi** | Nhanh, vì xử lý ngay trên máy người dùng, không cần qua mạng. | Chậm hơn (phải gửi yêu cầu lên server, server xử lý rồi gửi về). |
| **Bảo mật** | Thấp (mã nguồn dễ lộ, dễ bị giả mạo). | Cao (mã nguồn được giữ bí mật trên server). |
| **Phụ thuộc** | Vào trình duyệt và phiên bản trình duyệt. | Vào máy chủ web và phiên bản ngôn ngữ script trên server. |
| **Giao tiếp** | Với máy chủ thông qua AJAX/Fetch API (nếu cần dữ liệu động). | Giao tiếp với Database, các dịch vụ backend khác trên server. |
| **Kết quả trả về** | Không có kết quả trả về cho máy chủ (trừ khi dùng AJAX). | Trả về một trang HTML, CSS, JavaScript (có thể đã được tạo động) cho trình duyệt. |

Xuất sang Trang tính

**Tóm tắt luồng hoạt động tích hợp:**

1. Người dùng yêu cầu một trang web (.php).
2. Yêu cầu đến **Server Web**.
3. Server Web nhận yêu cầu, phát hiện đây là file PHP, chuyển nó cho **PHP Interpreter**.
4. PHP Interpreter thực thi mã PHP. Trong quá trình này:
   * Nó có thể kết nối với **Database** để lấy hoặc lưu dữ liệu.
   * Xử lý các logic nghiệp vụ.
   * Tạo ra một chuỗi HTML, CSS, và JavaScript (client-side script) dựa trên kết quả xử lý.
5. PHP Interpreter trả lại chuỗi HTML hoàn chỉnh đó cho Server Web.
6. Server Web gửi chuỗi HTML này về **Browser** của người dùng.
7. Browser nhận được HTML, phân tích và hiển thị trang. Nếu có client-side JavaScript, nó sẽ được **Browser thực thi** để tạo tương tác.

#### 4.3 Kiến thức về sử dụng PHP (Cú pháp, Kiểu dữ liệu, Biến, Hàm...)

**a. Cú pháp cơ bản:**

* **Dấu mở/đóng PHP:** Mã PHP phải nằm trong các cặp thẻ <?php và ?>.

PHP

<?php

// Mã PHP của bạn ở đây

echo "Hello, World!";?>

* **Thẻ rút gọn (short tags):** <? ... ?> (không khuyến khích sử dụng, cần cấu hình server).
* **Dấu chấm phẩy (;):** Kết thúc mỗi câu lệnh PHP.
* **Chú thích (Comments):**
  + Một dòng: // Đây là chú thích một dòng hoặc # Đây là chú thích một dòng
  + Nhiều dòng: /\* Đây là chú thích nhiều dòng \*/

**b. Biến (Variables):**

* Bắt đầu bằng dấu $.
* PHP là ngôn ngữ kiểu động, không cần khai báo kiểu dữ liệu.

$name = "Alice"; //

string$age = 30; // integer

$price = 19.99; // float

$isStudent = true; // boolean

**c. Kiểu dữ liệu (Data Types):**

* **Scalar types (Kiểu vô hướng):**
  + string: Chuỗi ký tự (sử dụng dấu nháy đơn ' hoặc nháy kép ").
  + integer: Số nguyên.
  + float (hoặc double): Số thực (có dấu phẩy động).
  + boolean: true hoặc false.
* **Compound types (Kiểu phức hợp):**
  + array: Mảng (tập hợp có thứ tự các giá trị, có thể chứa nhiều kiểu dữ liệu khác nhau).
  + object: Đối tượng (thể hiện của một lớp).
* **Special types (Kiểu đặc biệt):**
  + resource: Tài nguyên bên ngoài (ví dụ: kết nối cơ sở dữ liệu, file handle).
  + NULL: Biến không có giá trị (chưa được gán, hoặc được gán NULL).

**d. Toán tử (Operators):**

* **Số học:** +, -, \*, /, % (chia lấy dư), \*\* (lũy thừa).
* **Gán:** =, +=, -=, \*=, /=.
* **So sánh:** == (giá trị), === (giá trị và kiểu), !=, !==, >, <, >=, <=.
* **Logic:** && (AND), || (OR), ! (NOT), and, or, xor.
* **Nối chuỗi:** . (dấu chấm).

$greeting = "Hello" . " World!"; // "Hello World!"

**e. Cấu trúc điều khiển (Control Structures):**

* **Câu lệnh điều kiện:** if, else if, else

$score = 85;if ($score >= 90) {

echo "Excellent!";

} elseif ($score >= 70) {

echo "Good!";

} else {

echo "Needs improvement.";

}

* **Vòng lặp:** for, while, do...while, foreach (cho mảng)

for ($i = 0; $i < 5; $i++)

{

echo $i . " "; // 0 1 2 3 4

}

$fruits = ["apple", "banana", "cherry"];

foreach ($fruits as $fruit) {

echo $fruit . "<br>";

}

* switch**:**

$day = "Monday";switch ($day) {

case "Monday":

echo "It's Monday!";

break;

case "Friday":

echo "It's Friday!";

break;

default:

echo "Some other day.";

}

**f. Hàm (Functions):**

* Là các khối mã có thể tái sử dụng.

function greet($name) {

return "Hello, " . $name . "!";

}

echo greet("Bob"); // Output: Hello, Bob!

#### 4.4 Mục đích và Cú pháp PHP

Đây là phần tóm tắt lại và nhấn mạnh những điểm đã nói ở trên:

**Mục đích chính của PHP:**

* + **Tạo ra các trang web động (Dynamic Web Pages):** Nội dung trang web có thể thay đổi dựa trên dữ liệu từ cơ sở dữ liệu, tương tác của người dùng, hoặc các điều kiện khác.
  + **Thu thập dữ liệu từ biểu mẫu HTML (Form Data Handling):** Xử lý dữ liệu do người dùng gửi qua các biểu mẫu (POST/GET).
  + **Tương tác với cơ sở dữ liệu:** Kết nối, truy vấn, thêm, sửa, xóa dữ liệu trong các hệ quản trị CSDL như MySQL.
  + **Quản lý phiên (Session Management):** Duy trì trạng thái người dùng giữa các lần truy cập.
  + **Tạo các API (Application Programming Interface):** Cung cấp dữ liệu cho các ứng dụng khác (ví dụ: ứng dụng di động, ứng dụng JavaScript).
  + **Xử lý file:** Đọc, ghi, xóa file trên server.
  + **Gửi email.**
  + **Kiểm soát truy cập người dùng.**

**Cú pháp PHP (nhắc lại):**

* + Mã PHP luôn được đặt trong cặp thẻ mở và đóng: <?php ... ?>.
  + Mỗi câu lệnh kết thúc bằng dấu chấm phẩy ;.
  + Biến bắt đầu bằng $.
  + Hàm và các cấu trúc điều khiển tương tự C/C++/Java về mặt cú pháp { } và ( ).

#### 4.5 PHP cơ bản và xử lý nhập xuất dữ liệu trên web

**a. Xuất dữ liệu ra trình duyệt:**

* echo**:** Dùng để xuất chuỗi, số, biến ra trình duyệt. Không có giá trị trả về.

PHP

<?phpecho "<h1>Welcome to my PHP page!</h1>";$productName = "Laptop";echo "<p>Product: " . $productName . "</p>";?>

* print**:** Tương tự echo, nhưng có giá trị trả về (1). Thường dùng echo vì nhanh hơn một chút.
* print\_r() **và** var\_dump()**:** Dùng để debug, hiển thị cấu trúc và giá trị của mảng, đối tượng, biến với đầy đủ thông tin về kiểu dữ liệu.

**b. Nhập dữ liệu từ người dùng (Form Data):**

PHP xử lý dữ liệu gửi từ biểu mẫu HTML bằng cách sử dụng các biến siêu toàn cục (superglobals): $\_GET và $\_POST.

$\_GET**:**

* + Dùng để thu thập dữ liệu từ form có method="get".
  + Dữ liệu được gửi qua URL dưới dạng các cặp key-value (query string).
  + Thích hợp cho các yêu cầu lấy dữ liệu (ví dụ: tìm kiếm, phân trang) vì dễ dàng bookmark hoặc chia sẻ URL.
  + **Không an toàn cho dữ liệu nhạy cảm** vì dữ liệu hiển thị trên URL.
  + **Giới hạn kích thước dữ liệu.**

<!-- end list -->

HTML

<form action="process.php" method="get">

Name: <input type="text" name="username"><br>

<input type="submit" value="Submit GET"></form>

PHP

// Trong file process.php:<?phpif (isset($\_GET['username'])) {

$name = $\_GET['username'];

echo "Hello, " . htmlspecialchars($name) . "!"; // Luôn dùng htmlspecialchars để ngăn XSS!

} else {

echo "No username provided.";

}?>

$\_POST**:**

* + Dùng để thu thập dữ liệu từ form có method="post".
  + Dữ liệu được gửi trong phần "body" của yêu cầu HTTP, không hiển thị trên URL.
  + Thích hợp cho việc gửi dữ liệu nhạy cảm (ví dụ: mật khẩu, thông tin cá nhân) hoặc dữ liệu lớn.
  + Không có giới hạn kích thước đáng kể.

<!-- end list -->

HTML

<form action="process.php" method="post">

Email: <input type="email" name="user\_email"><br>

Password: <input type="password" name="user\_password"><br>

<input type="submit" value="Submit POST"></form>

PHP

// Trong file process.php:<?phpif (isset($\_POST['user\_email']) && isset($\_POST['user\_password'])) {

$email = $\_POST['user\_email'];

$password = $\_POST['user\_password'];

echo "Email: " . htmlspecialchars($email) . "<br>";

echo "Password: " . htmlspecialchars($password) . "<br>"; // Mật khẩu không nên hiển thị trực tiếp

// Thực hiện xác thực, lưu vào DB, v.v.

} else {

echo "Please fill out the form.";

}?>

$\_REQUEST**:** Chứa cả dữ liệu từ $\_GET, $\_POST, và $\_COOKIE. Không khuyến khích sử dụng thường xuyên vì khó kiểm soát nguồn gốc dữ liệu.

**Lưu ý quan trọng về bảo mật khi xử lý nhập liệu:**

* **Luôn luôn lọc và làm sạch (sanitize) dữ liệu đầu vào:** Sử dụng các hàm như htmlspecialchars() để ngăn chặn tấn công XSS khi hiển thị dữ liệu người dùng lên trang.
* **Luôn luôn kiểm tra và xác thực (validate) dữ liệu đầu vào:** Đảm bảo dữ liệu đúng định dạng, đúng loại, và nằm trong phạm vi cho phép trước khi xử lý hoặc lưu vào cơ sở dữ liệu.

#### 4.6 PHP Session và Cookie

Đây là hai cơ chế quan trọng để duy trì trạng thái của người dùng (state management) trong môi trường web phi trạng thái (stateless).

**a. Cookie:**

* **Khái niệm:** Là các file văn bản nhỏ được **lưu trữ trên trình duyệt của người dùng** bởi máy chủ web.
* **Mục đích:**
  + Lưu trữ các cài đặt của người dùng (ví dụ: ngôn ngữ, chủ đề).
  + Ghi nhớ thông tin đăng nhập (chức năng "ghi nhớ tôi").
  + Theo dõi hành vi người dùng (tracking).
* **Cơ chế hoạt động:**
  1. Máy chủ gửi một cookie đến trình duyệt thông qua header HTTP Set-Cookie.
  2. Trình duyệt lưu trữ cookie đó.
  3. Mỗi lần trình duyệt gửi yêu cầu đến cùng một máy chủ (cùng tên miền), nó sẽ tự động gửi lại tất cả các cookie liên quan trong header HTTP Cookie.
  4. Máy chủ nhận lại cookie và sử dụng thông tin trong đó.
* **Hàm trong PHP:**
  + setcookie(name, value, expire, path, domain, secure, httponly): Dùng để tạo hoặc cập nhật một cookie. Phải được gọi trước khi bất kỳ output nào được gửi đến trình duyệt.
  + $\_COOKIE: Biến siêu toàn cục dùng để truy cập các cookie đã gửi từ trình duyệt.
* **Ví dụ:**

<?php

// Đặt cookie tên "username" với giá trị "JohnDoe", hết hạn sau 1 giờ

setcookie("username", "JohnDoe", time() + 3600, "/");

if (isset($\_COOKIE['username'])) {

echo "Welcome back, " . htmlspecialchars($\_COOKIE['username']) . "!";

} else {

echo "Hello, guest!";

}

?>

* **Lưu ý:** Cookie có thể bị vô hiệu hóa bởi người dùng, có giới hạn kích thước, và tiềm ẩn rủi ro bảo mật nếu không được sử dụng đúng cách (ví dụ: không lưu thông tin nhạy cảm trực tiếp).

**b. Session (Phiên làm việc):**

* **Khái niệm:** Là một cách để **lưu trữ thông tin người dùng trên máy chủ** trong suốt một phiên làm việc (session). Thông tin này không được lưu trên trình duyệt của người dùng, mà được liên kết với một ID phiên duy nhất.
* **Mục đích:** Duy trì trạng thái đăng nhập, giỏ hàng, thông tin tạm thời khác của người dùng trong suốt quá trình họ duyệt web.
* **Cơ chế hoạt động:**
  1. Khi người dùng truy cập trang web lần đầu (hoặc khi bắt đầu một phiên), máy chủ tạo ra một **ID phiên duy nhất** và lưu trữ nó.
  2. ID phiên này được gửi đến trình duyệt dưới dạng một **cookie đặc biệt** (thường tên là PHPSESSID).
  3. Trình duyệt lưu cookie ID phiên này.
  4. Mỗi lần người dùng gửi yêu cầu tiếp theo, trình duyệt gửi lại cookie ID phiên.
  5. Máy chủ nhận ID phiên, dùng nó để truy xuất các dữ liệu đã lưu trữ cho phiên đó.
* **Hàm trong PHP:**
  + session\_start(): **Bắt buộc phải gọi ở đầu mỗi trang PHP** (trước bất kỳ output nào) để bắt đầu hoặc tiếp tục một phiên.
  + $\_SESSION: Biến siêu toàn cục dạng mảng, dùng để lưu trữ và truy xuất dữ liệu phiên.
  + session\_destroy(): Hủy bỏ toàn bộ phiên (thường dùng khi đăng xuất).
  + unset($\_SESSION['key']): Xóa một biến cụ thể trong phiên.
* **Ví dụ:**

<?php

session\_start(); // Bắt đầu phiên làm việc

if (isset($\_SESSION['views'])) {

$\_SESSION['views'] = $\_SESSION['views'] + 1;

} else {

$\_SESSION['views'] = 1;

}

echo "Bạn đã xem trang này " . $\_SESSION['views'] . " lần trong phiên này.";

// Đăng nhập

if (isset($\_POST['login'])) {

// Giả sử xác thực thành công

$\_SESSION['user\_id'] = 123;

$\_SESSION['username'] = "admin";

echo "<p>Bạn đã đăng nhập thành công!</p>";

}

// Đăng xuất

if (isset($\_POST['logout'])) {

session\_unset(); // Xóa tất cả biến session

session\_destroy(); // Hủy session

echo "<p>Bạn đã đăng xuất.</p>";

}?>

<form method="post">

<input type="submit" name="login" value="Đăng nhập">

<input type="submit" name="logout" value="Đăng xuất">

</form>

* **Ưu điểm của Session so với Cookie:**
  + **Bảo mật hơn:** Dữ liệu thực sự được lưu trên server, không bị lộ ra client.
  + **Không giới hạn kích thước:** Có thể lưu trữ lượng lớn dữ liệu.
  + **Dễ dàng kiểm soát thời gian sống.**

#### 4.7 Lập trình Hướng đối tượng trong PHP (OOP in PHP)

OOP là một mô hình lập trình dựa trên các khái niệm về "đối tượng" (objects), bao gồm dữ liệu (attributes) và mã (methods). PHP hỗ trợ đầy đủ OOP từ phiên bản PHP 5 trở đi.

**Mục đích của OOP:**

* **Tái sử dụng mã (Reusability):** Tạo ra các thành phần có thể tái sử dụng.
* **Dễ quản lý (Maintainability):** Code được tổ chức tốt hơn, dễ đọc và bảo trì.
* **Mở rộng (Extensibility):** Dễ dàng thêm chức năng mới mà không làm hỏng code cũ.
* **Trừu tượng hóa (Abstraction):** Ẩn đi sự phức tạp bên trong, chỉ hiển thị những gì cần thiết.

**Các khái niệm cơ bản trong OOP PHP:**

1. Class(Lớp):
   * Một "khuôn mẫu" hoặc "bản thiết kế" để tạo ra các đối tượng.
   * Định nghĩa các thuộc tính (properties) và phương thức (methods) mà các đối tượng tạo từ lớp đó sẽ có.

<?php

class Car {

// Thuộc tính (Properties)

public $brand;

public $color;

private $engineStatus = "off"; // Thuộc tính private

// Phương thức khởi tạo (Constructor)

public function \_\_construct($brand, $color) {

$this->brand = $brand;

$this->color = $color;

}

// Phương thức (Methods)

public function startEngine() {

$this->engineStatus = "on";

return "Engine started!";

}

public function stopEngine() {

$this->engineStatus = "off";

return "Engine stopped!";

}

public function getEngineStatus() {

return $this->engineStatus;

}

}

?>

1. Object (Đối tượng):

· Một thể hiện (instance) cụ thể của một lớp.

· Được tạo ra bằng từ khóa new

<?php

// Tạo đối tượng từ class Car

$myCar = new Car("Toyota", "blue");

$yourCar = new Car("Honda", "red");

echo $myCar->brand; // Output: Toyota

echo $yourCar->color; // Output: red

echo $myCar->startEngine(); // Output: Engine started!

echo $myCar->getEngineStatus(); // Output: on

?>

3.**Properties (Thuộc tính):**

* Các biến bên trong một lớp, biểu diễn các đặc điểm của đối tượng.
* Khai báo với public, protected, private.

4. **Methods (Phương thức):**

* Các hàm bên trong một lớp, biểu diễn hành vi hoặc hành động của đối tượng.

5. **Visibility (Phạm vi truy cập):**

* public: Có thể truy cập từ bất cứ đâu.
* protected: Chỉ có thể truy cập từ bên trong lớp đó và các lớp con kế thừa.
* private: Chỉ có thể truy cập từ bên trong lớp đó.

6. $this **(Tự tham chiếu):**

· Một biến đặc biệt trong PHP, dùng để tham chiếu đến đối tượng hiện tại (đối tượng đang gọi phương thức hoặc truy cập thuộc tính).

7.**Constructor (**\_\_construct()**):**

· Một phương thức đặc biệt được tự động gọi khi một đối tượng mới được tạo ra. Dùng để khởi tạo các thuộc tính của đối tượng.

**Các nguyên lý chính của OOP (4 trụ cột):**

* **Encapsulation (Đóng gói):** Gói dữ liệu (thuộc tính) và các phương thức hoạt động trên dữ liệu đó vào trong một đơn vị duy nhất (lớp). Đồng thời, ẩn đi chi tiết cài đặt bên trong và chỉ để lộ giao diện công khai (public methods) để tương tác. (Sử dụng public, protected, private).
* **Inheritance (Kế thừa):** Cho phép một lớp (lớp con - child class) kế thừa các thuộc tính và phương thức từ một lớp khác (lớp cha - parent class), giúp tái sử dụng mã.

class ElectricCar extends Car { // ElectricCar kế thừa từ Car

public $batteryCapacity;

public function \_\_construct($brand, $color, $capacity) {

parent::\_\_construct($brand, $color); // Gọi constructor của lớp cha

$this->batteryCapacity = $capacity;

}

public function charge() {

return "Charging battery...";

}

}

$tesla = new ElectricCar("Tesla", "black", "100kWh");echo $tesla->brand; // Từ lớp Carecho $tesla->charge(); // Từ lớp ElectricCar

* **Polymorphism (Đa hình):** Khả năng các đối tượng của các lớp khác nhau (liên quan đến kế thừa) phản ứng khác nhau với cùng một thông điệp (tức là cùng tên phương thức). Thường đạt được thông qua ghi đè phương thức (method overriding) hoặc Interface.
* **Abstraction (Trừu tượng hóa):** Tập trung vào những gì một đối tượng làm, thay vì cách nó làm. Sử dụng các lớp trừu tượng (abstract classes) và giao diện (interfaces) để định nghĩa các phương thức cần có mà không cung cấp cài đặt chi tiết, để các lớp con tự cài đặt.

**5. Kiến thức cơ bản lập trình với CSDL: PHP & mySQL**

**5.1 Khái niệm lập trình với CSDL.**

· **CSDL (Database) là gì?** CSDL là một tập hợp có tổ chức của dữ liệu, được lưu trữ và quản lý điện tử. Mục đích chính là lưu trữ, truy xuất, cập nhật và quản lý một lượng lớn thông tin một cách hiệu quả và có hệ thống.

· **Hệ quản trị CSDL (DBMS - Database Management System):** Là phần mềm giúp người dùng và các ứng dụng tương tác với CSDL. Nó cung cấp các công cụ để tạo, quản lý, truy vấn và bảo trì CSDL (ví dụ: MySQL, SQL Server, PostgreSQL, Oracle).

· **Lập trình với CSDL là gì?** Là việc sử dụng một ngôn ngữ lập trình (như PHP) để viết các đoạn mã giao tiếp với Hệ quản trị CSDL.

* **Mục đích:**
  + **Lưu trữ dữ liệu:** Ghi thông tin từ ứng dụng web vào CSDL (ví dụ: đăng ký người dùng, thêm sản phẩm).
  + **Truy xuất dữ liệu:** Lấy thông tin từ CSDL để hiển thị trên trang web (ví dụ: danh sách bài viết, thông tin cá nhân).
  + **Cập nhật dữ liệu:** Thay đổi thông tin đã có trong CSDL (ví dụ: chỉnh sửa hồ sơ người dùng).
  + **Xóa dữ liệu:** Loại bỏ thông tin khỏi CSDL.

· **Ngôn ngữ truy vấn CSDL:** SQL (Structured Query Language) là ngôn ngữ chuẩn để giao tiếp với hầu hết các CSDL quan hệ. Mặc dù bạn lập trình bằng PHP, nhưng các lệnh gửi đến CSDL vẫn phải là câu lệnh SQL.

5.2 Khái niệm cơ bản phân biệt CSDL quan hệ. Vd: mySQL và SQL Server.

**CSDL quan hệ (RDBMS - Relational Database Management System):** Là loại CSDL phổ biến nhất, tổ chức dữ liệu thành các bảng (tables). Mỗi bảng bao gồm các hàng (rows/records) và cột (columns/fields). Các bảng này có thể được liên kết với nhau thông qua các mối quan hệ (relationships) bằng cách sử dụng khóa chính (Primary Key - PK) và khóa ngoại (Foreign Key - FK).

**Đặc điểm chính của CSDL quan hệ:**

* **Bảng (Table/Relation):** Chứa dữ liệu về một thực thể cụ thể (ví dụ: bảng Users, bảng Products).
* **Hàng (Row/Record/Tuple):** Một bản ghi dữ liệu duy nhất trong bảng (ví dụ: một người dùng, một sản phẩm).
* **Cột (Column/Field/Attribute):** Một thuộc tính hoặc trường dữ liệu cụ thể trong bảng (ví dụ: username, email, product\_name).
* **Khóa chính (Primary Key - PK):** Một hoặc nhiều cột có giá trị duy nhất trong mỗi hàng, dùng để nhận diện duy nhất một bản ghi. Không được phép trùng lặp hoặc NULL.
* **Khóa ngoại (Foreign Key - FK):** Một cột trong một bảng mà giá trị của nó tham chiếu đến khóa chính của một bảng khác. Khóa ngoại thiết lập mối quan hệ giữa các bảng.
* **SQL:** Ngôn ngữ chuẩn để truy vấn và thao tác dữ liệu.

### ✅ ****So sánh MySQL vs SQL Server****

| **Tiêu chí** | **MySQL** | **SQL Server** |
| --- | --- | --- |
| Hãng phát triển | Oracle | Microsoft |
| Môi trường phổ biến | Linux, PHP, Web (open-source) | Windows, .NET |
| Ngôn ngữ truy vấn | SQL chuẩn | SQL chuẩn + T-SQL mở rộng |
| Giao diện quản lý | phpMyAdmin, MySQL Workbench | SQL Server Management Studio |

5.3 Kiến thức cơ bản lập trình PHP & mySQL. Vd: các lỗi thường gặp khi kết nối CSDL, kiểu,recordset, đóng kết nối khi thao tác xong...

### 1. ****Kết nối MySQL trong PHP****

**<?php**

**$servername = "localhost";**

**$username = "root"; // Tên người dùng CSDL của bạn**

**$password = ""; // Mật khẩu CSDL của bạn (thường trống với XAMPP/WAMP)**

**$dbname = "your\_database"; // Tên CSDL của bạn**

**// Tạo kết nối**

**$conn = mysqli\_connect($servername, $username, $password, $dbname);**

**// Kiểm tra kết nối**

**if (!$conn) {**

**die("Kết nối CSDL thất bại: " . mysqli\_connect\_error());**

**}**

**echo "Kết nối CSDL thành công!";**

**?>**

2.**Thực thi truy vấn SQL:**

* Sử dụng hàm mysqli\_query() để gửi câu lệnh SQL đến CSDL.
* Kết quả của mysqli\_query() phụ thuộc vào loại câu lệnh SQL:
  + SELECT: Trả về một **result set object** (hay còn gọi là **recordset**).
  + INSERT, UPDATE, DELETE: Trả về TRUE nếu thành công, FALSE nếu thất bại.

3.**Xử lý Recordset (Kết quả truy vấn SELECT):**

* Nếu câu lệnh là SELECT, bạn cần lấy từng hàng dữ liệu từ recordset.
* Các hàm phổ biến:
  + mysqli\_fetch\_assoc($result): Lấy một hàng dữ liệu dưới dạng mảng kết hợp (associative array), nơi các khóa mảng là tên cột.
  + mysqli\_fetch\_row($result): Lấy một hàng dữ liệu dưới dạng mảng số (numeric array).
  + mysqli\_num\_rows($result): Trả về số lượng hàng trong recordset.

4.**Đóng kết nối khi thao tác xong:**

* Sử dụng hàm mysqli\_close($conn) để đóng kết nối CSDL.
* Điều này giúp giải phóng tài nguyên trên máy chủ CSDL. Mặc dù PHP tự động đóng kết nối khi script kết thúc, việc đóng tường minh là một thói quen tốt, đặc biệt trong các ứng dụng lớn.

**Các lỗi thường gặp khi kết nối CSDL:**

* **Sai tên server, username, password, hoặc database name:** Kiểm tra lại thông tin đăng nhập CSDL.
* **MySQL server chưa chạy:** Đảm bảo dịch vụ MySQL đã được khởi động (ví dụ: trong XAMPP/WAMP).
* **Lỗi phân quyền truy cập:** Người dùng CSDL không có quyền truy cập vào database hoặc thực hiện các thao tác mong muốn.
* **Cổng (port) không đúng:** Mặc định MySQL là cổng 3306.
* **Lỗi cú pháp SQL:** Câu lệnh SQL bạn gửi không đúng.
* **Không chọn đúng database:** Nếu bạn kết nối mà không chỉ định database, bạn phải dùng mysqli\_select\_db() sau đó. (Nhưng tốt nhất nên chỉ định ngay trong mysqli\_connect()).

**5.4 Truy xuất CSDL bằng PHP**

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "your\_database";

$conn = mysqli\_connect($servername, $username, $password, $dbname);

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

$sql = "SELECT id, fullname, email FROM users"; // Câu lệnh SQL SELECT

$result = mysqli\_query($conn, $sql); // Thực thi truy vấn

if (mysqli\_num\_rows($result) > 0) {

// Xuất dữ liệu của mỗi hàng

while($row = mysqli\_fetch\_assoc($result)) {

echo "ID: " . $row["id"]. " - Name: " . $row["fullname"]. " - Email: " . $row["email"]. "<br>";

}

} else {

echo "0 results";

}

mysqli\_close($conn); // Đóng kết nối

?>  
**5.5 Kết hợp HTML Form và CSDL bằng PHP.**

HTML

<form method="post">

<input name="username" placeholder="Tên người dùng">

<input name="email" placeholder="Email">

<button type="submit">Thêm</button>

</form>

PHP

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$conn = mysqli\_connect("localhost", "root", "", "web");

$u = $\_POST["username"];

$e = $\_POST["email"];

$sql = "INSERT INTO users (username, email) VALUES ('$u', '$e')";

mysqli\_query($conn, $sql);

mysqli\_close($conn);

}

**5.6 Khái niệm cơ bản về SQL injection. Từ khóa: “php mysql SQL injection”**

**a. Khái niệm:**

* **SQL Injection (SQLi)** là một **lỗ hổng bảo mật web nghiêm trọng** cho phép kẻ tấn công thực thi các câu lệnh SQL độc hại vào CSDL thông qua các trường nhập liệu của ứng dụng web.
* Lỗ hổng này xảy ra khi ứng dụng không xử lý hoặc làm sạch (sanitize) đúng cách dữ liệu đầu vào từ người dùng trước khi sử dụng nó để xây dựng các câu lệnh SQL.
* Kẻ tấn công có thể "chèn" (inject) các phần của câu lệnh SQL vào dữ liệu đầu vào, làm thay đổi ý định ban đầu của câu lệnh SQL được thực thi.
* Ví dụ tấn công:  
  SELECT \* FROM users WHERE username='' OR 1=1

### ✅ ****Cách phòng chống SQL Injection trong PHP****

**$conn = mysqli\_connect("localhost", "root", "", "web");**

**$stmt = mysqli\_prepare($conn, "INSERT INTO users (username, email) VALUES (?, ?)");**

**mysqli\_stmt\_bind\_param($stmt, "ss", $u, $e);**

**$u = $\_POST["username"];**

**$e = $\_POST["email"];**

**mysqli\_stmt\_execute($stmt);**